A Library of Analytic Collision Geometry Functions for Real-Time Shortest Distance and Proximity Computation

João Afonso[1], Raquel Agostinho[1], Inês Lúcio[1]

Abstract
One of the most important aspects that should be considered in the interaction between two objects is their collision state. In Biomedical Engineering, Contact Detection calculations appear in movement simulations and virtual reality medical applications, for example. While several Contact Detection algorithms have been formulated so far, there is still a lack of a curated collection of analytical approaches that are scattered within the body of literature. Thus, in this paper, we (i) conducted a literature review on analytical Collision Detection and Proximity Query techniques; (ii) implemented several contact pair methods in C# scripts that will make part of a more extensive systematic library of functions. To illustrate the functioning of the library, we developed an interactive application in Unity to perform real-time Minimum Distance and Collision Calculations. A simple foot-ground biomechanical simulation was also designed to test the implemented methods. Our work contributes to the mathematical description of real-world collisions and allows us to perform simulations in various fields.

Keywords
Closest Distance — Proximity Query — Collision Detection — Unity — Biomechanics — C#

[1] *Biomedical Engineer student , Instituto Superior Técnico, Lisboa, Portugal*

Contents

Introduction

When studying the interaction between two bodies, one of the most important aspects to consider is their collision state, i.e., whether the bodies intersect or not at a certain point or region. Collision Detection allows us to understand the mechanical behavior of bodies, from where its study is essential and a subject of common interest in many areas.

For this purpose, to find if whether two objects intersect, it is necessary to compute the Minimum Distance between them or to verify if a separation condition holds or not. For Contact Detection, the value of the Closest Distance may be used to represent the state of collision between objects. Therefore, when two bodies are separated, the Minimum Distance between them is positive; when there is contact at a single point, the distance becomes zero; when they overlap, the distance becomes negative. In the alternative, Contact Detection can be computed using Proximity Queries, which report information regarding objects' relative placement or geometry. Common examples include checking whether two bodies are overlapping in space or if their boundaries intersect [1].

These calculations are inherently related to the bodies' geometry, thus it's crucial to choose a geometry that not only faithfully represents the object but also allows efficient and effective calculations.

Collision Detection has many applications in a variety of fields, including video games development, digital-rapid prototyping, and robotic simulations [2]. In Biomedical Engineering, Contact Detection can be used in motion simulations that study foot-to-ground contact [3] and in medical applications that rely on virtual reality [4].

Although several Collision Detection algorithms [5, 6, 7] have already been formulated, most of which consist of numerical approaches, there is still a need for an organized compilation of analytical methods and an improvement in the efficiency of the existing ones.

To meet these needs, in this paper, we propose performing a literature review and compiling analytical solutions in a systematized and parameterized library translatable to other programming languages. To demonstrate how the library works, we proposed the development of an interactive application in Unity that uses the implemented code to compute Minimum Distances and Proximity Queries in real-time. We also present a contact simulation between a foot abstraction and the ground as one of the library's possible biomechanical applications.

Furthermore, the main focus of this paper is to (i) perform a literature review on Collision Detection analytical methods; (ii) implement some methods in C# scripts that will make part of a larger systematic library of functions, and (iii) develop a simple biomechanical simulation to test several of the implemented Collision Detection methods. Moreover, a description of the geometric primitives used and an illustration of the calculations for detecting collision is also presented.

1. Literature Review

First, we conducted an extensive literature review and compiled all the references found into four tables. The literature review started with the most significant textbooks and reports on the topics. Then, we extended the revision to papers on more complex geometric pairs. Lastly, the review ended with source code hosting platforms such as GitHub to facilitate future implementation of the calculations.

2. Methods

To reach the proposed goals, first, the subject was divided: calculation of the Closest Distance and Proximity Query. Then, we made further subdivisions concerning the geometry of the objects, which divided them into two and three-dimensions, respectively.

Following the literature review, a selection of simple Collision Detection pairs was conducted to kick-start what will become a library with all the known analytical Collision Detection solutions. Finally, we developed an interactive application using Unity to display the library's functionality. In addition, a biomechanical simulation was also designed in Unity to demonstrate the library's applicability in Biomechanics.

2.1 Geometric primitives

Defining a 2D or 3D object is central before implementing Minimum Distance and Proximity Query algorithms. Well-known formulas of several objects' representations are listed and detailed in the following subsections.

2.1.1 Point

In geometry, a point is a location represented by a dot. It has no length, width, shape, or size; it only has its specific position. Therefore, to define a point in a 2D space, it is only necessary to know it's x and y coordinates. In a 3D referential, an additional coordinate, z, must be known.

2.1.2 Line Segment

Although there are various ways to represent a line segment, in this paper, we'll use the parametric form to make coherent the definition with its use in Closest Distance calculations and Proximity Queries. Thus, a generic line is represented as

$$L(t) = P + t\vec{v},\ \text{for } t \in \mathbb{R} \tag{1}$$

being P a point of the line.

A line segment, or simply segment, is a line with the parametric restriction $t \in [t_0, t_1]$. If P_0 and P_1 are endpoints of the segment and $\vec{d} = P_1 - P_0$, the same equation is suitable as Equation (1):

$$L(t) = P + t\vec{d},\ \text{for } t \in [0, 1] \tag{2}$$

2.1.3 Circle/Sphere

A circle representation consists of the definition of its coordinates in relation to its radius $r > 0$. In its implicit form, the center point $O(a, b)$ is used to define the following equation:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{3}$$

The transposition of a 2D circle to a 3D representation defines a sphere. On the same thought process, an additional coordinate is added, and the center point $O(a, b, c)$ sets the similar equation:

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2 \tag{4}$$

2.1.4 Axis Aligned Bounding Box (AABB)

A rectangle in two-dimensional space, or a bounding box, is said to be axis-aligned if the edge vectors are parallel to the coordinate axes. Therefore, its representation uses two edge vectors, $\vec{e}_0$ and $\vec{e}_1$, that are perpendicular and aligned with the referential axes. Figure 1 shows the representation of an AABB, in which P, the origin (0,0), is the minimum point of the AABB, and the vectors t_0 and t_1 are parallel to the X and Y axis, respectively.

$$X(t_0, t_1) = P + t_0\vec{e}_0 + t_1\vec{e}_1 \text{ for } t_0 \in [0, 1] \text{ and } t_1 \in [0, 1] \tag{5}$$

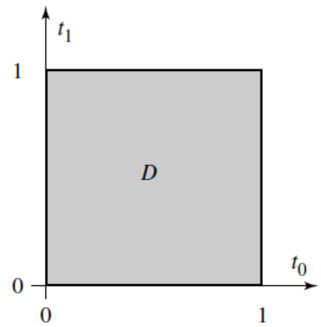

Figure 1. Parametric representation of an AABB, where D represents the domain of the geometric form. [8]

A 3D box is a rectangular parallelepiped whose faces are each perpendicular to one of the basis vectors. In the desired context, an AABB solely needs its maximum and minimum point to be defined and to be able to calculate Closest Distances and Proximity Queries. Therefore, $P_{min} = [x_{min}, y_{min}, z_{min}]$ and $P_{max} = [x_{max}, y_{max}, z_{max}]$, can be used to describe regions of space between two planes, allowing, in a simple way, the detection of Closest Distances.

2.1.5 Oriented Bounding Box (OBB)

Although all rectangles can be said to be oriented, an oriented rectangle is a rectangle not aligned with the main axis, therefore, the use of axis-aligned vectors is irrelevant. The parametric form uses the symmetric representation of an oriented bounding box, which consists of a center point P, two unit-length vectors $\hat{u_0}$ and $\hat{u_1}$ that are perpendicular, and two extents $e_0 > 0$ and $e_1 > 0$, as seen in Figure 2. Its parametrization is the following:

$$X(t_0, t_1) = P + t_0\hat{u_0} + t_1\hat{u_1} \text{ for } |t_0| \leq e_0 \text{ and } |t_1| \leq e_1. \quad (6)$$

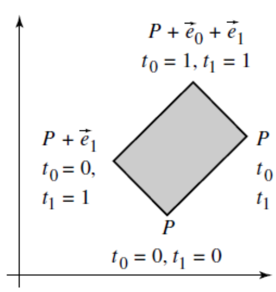

Figure 2. Parametric representation of an OBB. [8]

As an OBB is simply an AABB with a different orientation, the method used to describe an OBB is similar to the one described previously in Section 2.1.4.

2.2 Closest distance between primitives

This subsection explores several analytical formulas that describe the Minimum Distance between two primitives, along with some more straightforward implementations used to calculate this distance.

2.2.1 Two Points

The Closest Distance between any two points can be computed by the Euclidean distance, which represents the length of the line connecting the points. This distance is a direct result of the Pythagorean theorem. Given two points, A and B, in two-dimensional space with the coordinates:

- A = (x_1, y_1)
- B = (x_2, y_2)

To find the distance between them, d, the Pythagorean theorem can be rewritten as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (7)$$

In three dimensions, an additional coordinate, z, has to be added to the equation, which takes the form of:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (8)$$

2.2.2 Point and Line Segment

Given a point Y and a line segment L parameterized by (2), the Minimum Distance between the two corresponds to the distance between the point Y and its orthogonal projection Y' onto the line segment, for some parameter value t'. This definition is valid for two or three-dimensional spaces. However, Y' may not land directly on the line segment L. It may lie behind the origin or in front of the segment's endpoint. Figure 3 shows the different possibilities.

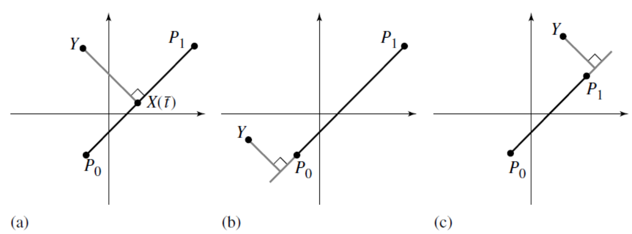

Figure 3. Nearest point of a line segment to a given point. [8]

Thus, it is first necessary to calculate t' as one would when calculating the closest distance between a point and a line. Due to the vector $\overrightarrow{(Y - Y')}$ being perpendicular to the direction of the line, $\vec{v}$, then:

$$t' = \vec{v} \cdot \frac{(Y - P)}{||\vec{v}||^2} \quad (9)$$

From the above entities, it follows that the intended distance is:

$$d = ||Y - P - t'\vec{v}|| = \sqrt{||Y - P||^2 - \frac{(\vec{v} \cdot (Y - P))^2}{||\vec{v}||^2}} \quad (10)$$

Then, the value must be checked against the interval $[0,1]$, and the final Minimum Distance is given by:

$$\textbf{Closest Distance}: \begin{cases} \|Y-P\|, & \text{if } t' \leq 0 \\ \|Y-(P+t'\vec{v})\|, & \text{if } 0 < t' < 1 \\ \|Y-(P+\vec{v})\|, & \text{if } t' \geq 1 \end{cases} \quad (11)$$

The first case corresponds to when Y' falls behind the origin of the line segment. The second one corresponds to when Y' falls on the line segment, while the third one is when Y' falls over the end of the line segment.

2.2.3 Point and Circle/Sphere

A simple way to calculate the closest distance between a point and a circle or sphere is by taking into account the distance between a given point and the center of the said circle/sphere. Considering a point P and a circle C or a sphere S, with its center O and its radius r, respectively, the distance between P and O, d, can be obtained by the Euclidean distance as described in Section 2.2.1. Then, to compute the Minimum Distance, it is necessary to subtract the radius:

$$\textbf{Closest Distance}: d - r \quad (12)$$

2.2.4 Two Line Segments

Considering the segments $L_i(t) = P_i + t_i\vec{d}_i$, for $i = 0,1$ and $t \in [0, T_i]$, the points of interest $(\bar{t}_0, \bar{t}_1)$, the zeros of the function F described in Equation (13), and the minimum points of F: $(\hat{t}_0, 0)$, $(0, \hat{t}_1)$,

$$F(t_0, t_1) = \left\| t_0\vec{d}_0 - t_1\vec{d}_1 + \vec{\Delta} \right\|^2, \text{with } \vec{\Delta} = P_0 - P_1 \quad (13)$$

Before computing the Minimum Distance between two line segments, it is first necessary to determine their cross-product to establish whether they are parallel or not. Once done, if the lines are non-parallel and if $(\bar{t}_0, \bar{t}_1) \in (0, T_0) \times (0, T_1)$, the segments intersect at inner points. If not, it should be inferred where the level curves of function F centered at t $(\bar{t}_0, \bar{t}_1)$ first satisfy the domain boundaries.

In short, for the non-parallel case:

$$\textbf{Closest Distance}: \begin{cases} 0, \text{ if } \bar{t}_0 \in (0,T_0) \text{ and } \bar{t}_1 \in (0,T_1) \\ \frac{\left|\vec{d}_0^{\perp}\cdot\vec{\Delta}\right|}{\left\|\vec{d}_0\right\|}, \text{ if } \hat{t}_0 \in (0,T_0) \text{ and } \bar{t}_1 \leq 0 \\ \frac{\left|\vec{d}_0^{\perp}\cdot(\vec{\Delta}-T_1\vec{d}_1)\right|}{\left\|\vec{d}_0\right\|}, \text{ if } \hat{t}_0 \in (0,T_0) \text{ and } \bar{t}_1 \geq T_1 \\ \frac{\left|\vec{d}_1^{\perp}\cdot\vec{\Delta}\right|}{\left\|\vec{d}_1\right\|}, \text{ if } \hat{t}_1 \in (0,T_1) \text{ and } \bar{t}_0 \leq 0 \\ \frac{\left|\vec{d}_1^{\perp}\cdot(\vec{\Delta}+T_0\vec{d}_0)\right|}{\left\|\vec{d}_1\right\|}, \text{ if } \hat{t}_1 \in (0,T_1) \text{ and } \bar{t}_0 \geq T_0 \\ \left\|\vec{\Delta}\right\|, \text{ if } \hat{t}_0 \leq 0 \text{ and } \hat{t}_1 \leq 0 \\ \left\|\vec{\Delta}+T_0\vec{d}_0\right\|, \text{ if } \hat{t}_0 \geq T_0 \text{ and } \hat{t}_1 \leq 0 \\ \left\|\vec{\Delta}-T_1\vec{d}_1\right\|, \text{ if } \hat{t}_0 \leq 0 \text{ and } \hat{t}_1 \geq T_1 \\ \left\|\vec{\Delta}+T_0\vec{d}_0-T_1\vec{d}_1\right\|, \text{ if } \hat{t}_0 \geq 0 \text{ and } \hat{t}_1 \geq T_1 \end{cases} \quad (14)$$

The Closest Distance between two line segments at their intersection is zero, as seen in the first example. The second case shows when a first segment interior point is closest to the second segment's origin. The third one shows when a first segment's inner point is most near the second segment's endpoint. The fourth expression should be applied when an interior point of the second segment and the first segment's origin are the closest. The fifth case should be employed when a second segment's inner point and the end of the first segment are the nearest points. The sixth and ninth examples show what happens when the segments' beginning and ending points are their closest points, respectively. The eighth scenario describes the condition when the start of the first segment and the end of the second segment are the nearest positions. The seventh case describes the contrary occurrence.

And for the parallel case:

$$\textbf{Cl. Dist.}: \begin{cases} \left\|\vec{\Delta}\right\|, \text{ if } \vec{d}_0\cdot\vec{d}_1 < 0 \text{ and } \vec{d}_0\cdot\vec{\Delta} \geq 0 \\ \left\|\vec{\Delta}+T_0\vec{d}_0\right\|, \text{ if } \vec{d}_0\cdot\vec{d}_1 > 0 \text{ and } \vec{d}_0\cdot(\vec{\Delta}+T_0\vec{d}_0) \geq 0 \\ \left\|\vec{\Delta}-T_1\vec{d}_1\right\|, \text{ if } \vec{d}_0\cdot\vec{d}_1 > 0 \text{ and } \vec{d}_0\cdot(\vec{\Delta}-T_1\vec{d}_1) \geq 0 \\ \left\|\vec{\Delta}+T_0\vec{d}_0-T_1\vec{d}_1\right\|, \text{ if } \vec{d}_0\cdot\vec{d}_1 < 0 \text{ and } \vec{d}_0\cdot(\vec{\Delta}+ \\ +T_0\vec{d}_0-T_1\vec{d}_1) \geq 0 \\ \frac{\left|\vec{d}_0^{\perp}\cdot\vec{\Delta}\right|}{\left\|\vec{d}_0\right\|}, \text{ otherwise} \end{cases} \quad (15)$$

In this case, the latter description is when the projection of one segment onto another intersects that same segment. The first four cases represent the same cases as the last four in the previous situation.

2.2.5 Line Segment and Circle/Sphere

Concisely, calculating the Closest Distance between a line segment and a circle is to calculate the distance between the center of the circle to the line minus the circle radius. Therefore, considering a circle with radius r, and using Equation (11), the Closest Distance between a line segment and a circle is given by:

$$\textbf{Closest Distance}: \begin{cases} \|Y-P\|-r, & \text{if } t' \leq 0 \\ \|Y-(P+t'\vec{v})\|-r, & \text{if } 0 < t' < 1 \\ \|Y-(P+\vec{v})\|-r, & \text{if } t' \geq 1 \end{cases} \quad (16)$$

The cases in Equation (16) are the same as described in Section 2.2.2.

2.2.6 Two Circles/Spheres

To obtain the minimum distance between two circles, C_a and C_b, respectively, one must consider its centers, O_a and O_b, and its radii, r_a and r_b. The distance between the two centers, d, can be calculated by the Euclidean distance, as shown in Section 2.2.1, and once both radii are subtracted from this distance, we get the Closest Distance:

$$\textbf{Closest Distance} : d - r_a - r_b \tag{17}$$

Analogous calculations can be carried in 3D, using two spheres, S_a and S_b, respectively, and its centers and radii. The Closest Distance can be computed by the same Equation (17) described above.

2.3 Proximity Queries between primitives

In this subsection, other analytic formulas are presented describing the collision state between two bodies without the requirement to determine the smallest distance between them. Additionally, some simplified solutions are offered as well.

2.3.1 Two AABBs

Two tests (one for each axis) are required to determine whether two AABBs, A and B, respectively, are colliding in 2D. For example, when performing the test on the X-axis, we must check if the ranges A_{min_X}-A_{max_X} and B_{min_X}-B_{max_X} overlap. A similar test can be performed on the Y-axis and, when considering the third dimension (3D), on the Z-axis. Mathematically, the following conditions need to be verified:

$$\textbf{Proximity Query} : \begin{cases} (A_{min_X} \leq B_{max_X}) \wedge (A_{max_X} \geq B_{min_X}) \\ (A_{min_Y} \leq B_{max_Y}) \wedge (A_{max_Y} \geq B_{min_Y}) \\ (A_{min_Z} \leq B_{max_Z}) \wedge (A_{max_Z} \geq B_{min_Z}) \end{cases} \tag{18}$$

2.3.2 Two OBBs

One of several existing methods to verify OBB-OBB intersection is the Separating Axes Theorem, SAT. If it is feasible to identify an axis - on which the geometry of each object is projected - where these projections do not overlap, then two convex bodies are said not to collide according to SAT.

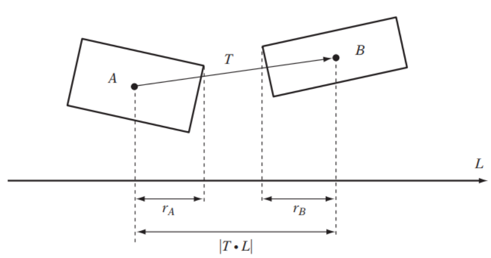

Figure 4. A SAT test applied to two bounding boxes in 2D. Since the sum of their projected extents $(r_a + r_b)$ is less than the distance between their projected centers $(\|T \cdot L\|)$, the objects do not overlap. [9]

The normal vector perpendicular to the objects' edges (in two dimensions) or faces (in three dimensions) determines the used axes' orientation. In 2D, due to symmetry, only four separation axes need to be tested, two for each object, corresponding to the normals of the edges. In three dimensions, 15 cases need to be considered:

- Three face normals from the first object;
- Three face normals from the second object;
- The nine normals formed by computing the cross-product between the normals above.

For example, given two OBBs, A and B, the SAT tests would involve projecting the vector joining the centers of two boxes, $d_{AB} = c_A - c_B$, onto the chosen separation axis, L. Then, it is checked whether the sum of the extensions of the projected boxes, $(\|h_A \cdot L\|) + (\|h_B \cdot L\|)$, exceed d_{AB}. There is no intersection, and testing can cease if an axis is located where this does not occur. On the other hand, contact is present if, for all of the axes examined, the projection of the box extensions is greater than the projection of dAB.

In a nutshell, given:

$$t = \|d_{AB} \cdot L\| - (\|h_A \cdot L\| + \|h_B \cdot L\|) \tag{19}$$

Then, the Proximity Query is given by:

$$\textbf{Proximity Query} : \begin{cases} \exists L, t > 0 \Rightarrow \textbf{No collision} \\ \forall L, t \leq 0 \Rightarrow \textbf{Collision} \end{cases} \tag{20}$$

2.3.3 Point and Sphere

The intersection of a point $P(x_P, y_P, z_P)$ and a sphere S happens when that sphere contains the point. In other words, there is a need to check if the distance between the point and the center of the sphere $O(x_S, y_S, z_S)$ is smaller than or equal to the sphere's radius, r. The distance between P and O can be computed through the Formula (8) presented in Section 2.2.1 and to conclude if the point is inside the sphere, we must assess the following condition:

$$\textbf{Proximity Query} : \sqrt{(x_P - x_S)^2 + (y_P - y_S)^2 + (z_P - z_S)^2} \leq r \tag{21}$$

2.3.4 Point and AABB

To verify if a point P and an AABB A intersect, one must check if the point coordinates fall inside the AABB. More precisely, it can be attained by validating the following conditions:

$$\textbf{Proximity Query} : \begin{cases} (P_X \leq A_{max_X}) \wedge (P_X \geq A_{min_X}) \\ (P_Y \leq A_{max_Y}) \wedge (P_Y \geq A_{min_Y}) \\ (P_Z \leq A_{max_Z}) \wedge (P_Z \geq A_{min_Z}) \end{cases} \tag{22}$$

2.3.5 Sphere and AABB

There are a few ways as one can verify if a sphere S and an AABB A are colliding. One approach can be evaluating every vertex of the AABB and doing a point and sphere test, as described in Section 2.3.3.

However, a simplified way of detecting this collision is by calculating the distance between the AABB's closest point P, which is not necessarily a vertex, and the sphere's center C and seeing if it is less than or equal to the sphere's radius, r. The distance between P and C, d, can be calculated using Euclidean distance as shown in Section 2.2.1. Thus, the condition that needs to be tested is:

$$\textbf{Proximity Query} : d \leq r \tag{23}$$

2.3.6 Two Spheres

Simply determining if the distance between the spheres' centers is less than or equal to the total of their radii will determine whether two spheres are intersecting. Thus, two spheres, S_A and S_B, with centers $O_A(x_a, y_a, z_a)$ and $O_B(x_b, y_b, z_b)$ and radii r_a and r_b, respectively, are in colliding if:

$$\textbf{Prox. Query} : (x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2 \leq (r_a + r_b)^2 \tag{24}$$

2.4 Library of Functions

The calculations in Sections 2.2 and 2.3 were converted into parameterized functions which were then later compiled into two C# scripts using Unity's 2021.3.3f1 version, one regarding the Minimum Distance calculations and the other regarding Proximity Queries.

2.5 Collision Detection Application

Using the Unity game development platform, it was possible to test the created function libraries interactively. With the previously described primitives and Unity game objects, we built a minimalist interactive application that is described in Section 3.3. These are based on the calculations of the Minimum Distance between two objects or their state of collision (Sections 2.3 and 2.2). The central purpose of this interactive application is to allow the user to explore the libraries with some graphical and visual support.

2.6 Biomechanical Simulation

To frame the work carried out in the area of Biomechanics, a simple animation of a robotic foot making contact with the ground was also developed. Again, we used the Unity platform to create this simulation, using some functions of the libraries to test the Minimum Distance and detect collision of the foot parts with the ground plane.

3. Results and Discussion

Following the methodology described in Section 2, we achieved some outcomes. From these, we highlight the tables produced as a result of the literature review and the function libraries created after the selection of some contact pairs. Then, using the libraries developed, an interactive application and a biomechanical simulation were designed using the Unity game engine.

3.1 Tables from the Literature Review

Regarding the literature review, in total, 30 different references were revised. Of these, five were books, 20 were papers, and five were online information repositories. Besides these 30 references, a review of another seven information repositories was also initiated, which given the research's extension, was not completed.

The developed tables refer to Minimum Distance calculations and Proximity Queries between different geometric pairs in two or three dimensions. Their structure is the following:

- A column for the geometric pair;
- A column for the references regarding that pair;
- A column for the used reference when implemented or the reference used if implementing (if blank, the reference has not retrieved any feasible functions nor equations to implement);
- A column for the specific equation used for the implementation calculation;

The Tables 1 and 2 are a simplified version of the tables mentioned above. The full version of the mentioned tables can be found in Appendix I that lists all 55 contact pairs for Closest Distance calculations, 25 in two dimensions and 30 in three dimensions, as well as 145 contact pairs for Proximity Queries, 59 in two and 86 in three dimensions.

The separation into four different tables was done to facilitate consultation, given the extent of the information. The internal division adopted allows anyone to understand the implementation made for each contact pair if they wish to recreate it.

3.2 Library of Functions

The two scripts that have been developed consist of a compilation of functions concerning Minimum Distance calculations or Proximity Queries between different primitives.

The function library for calculating the Closest Distance between contact pairs contains 12 functions, which account for the computation in two and three dimensions of six different contact pairs. Table 3 summarizes the appropriate uses of these functions. Regarding Collision Detection, the library developed contains eight functions. Similarly, Table 4 summarizes the appropriate use of these functions.

3.3 Collision Detection Application

As previously stated, as part of a Collision Detection API created in Unity, we applied the functions referred to in Section 3.2. All the equations (Eq. 1-24) were adapted to C#, and all the developed code was built to make the experience more pleasant for the user while also adding alluring proprieties to the game. We will address some of the interface's characteristics and all the in-game features in this subsection.

Table 1. Closest Distance Table.

Contact Pairs	References	Specific Location/Equation
Point - Point	[10]	Section "Distance formulas"
Point - Line Segment	[8]	Pages 127-129 and 367-369
Point - Circle/Sphere	[8]	Pages 388-391 and 401-403
Line Segment - Line Segment	[8]	Pages 228-229 and 415-418
Line Segment - Circle/Sphere	[8]	Pages 231-233
Circle - Circle/Sphere - Sphere	[10]; [8]	Section "Distance formulas"; Pages 388-391 and 401-403

Table 2. Proximity Queries Table.

Contact Pairs	References	Specific Location/Equation
AABB- AABB	[8]; [9]	Pages 635-637; Pages 161-164
OBB - OBB	[11]; [8]	Pages 35-37; Pages 639-644
Point - Sphere	[12]	Section "Point vs. sphere"
Point - AABB	[9]; [12]	Pages 130-132; Section "Point vs. AABB"
Sphere - AABB	[9]	Pages 165-166
Sphere - Sphere	[9]	Pages 88-89

Table 3. Closest Distance Function Library.

Closest Distance	
Contact Pair	**Inputs**
2D	
Point-Point	First point position, Second point position
Point-Line Segment	Point position, Line segment origin, Line segment end-point
Circle-Line Segment	Circle origin, Line segment origin, Line segment end-point, Circle radius
Line Segment- Line Segment	First line segment origin, First line segment end-point, Second line segment origin, Second line segment end-point
Point-Circle	Point position, Circle origin, Circle radius
Circle-Circle	First circle origin, Second circle origin, First circle radius, Second circle radius
3D	
Point-Point	First point position, Second point position
Point-Line Segment	Point position, Line segment origin, Line segment end-point
Line Segment- Line Segment	First line segment origin, First line segment end-point, Second line segment origin, Second line segment end-point
Point-Sphere	Point position, Sphere origin, Sphere radius
Sphere-Line Segment	Sphere origin, Line segment origin, Line segment end-point, Sphere radius
Sphere-Sphere	First sphere origin, Second sphere origin, First sphere radius, Second sphere radius

3.3.1 Interface

The developed interface has an initial menu where the user can choose between the two types of calculations implemented, Proximity Queries and Closest Distance. When one of the two buttons is pressed, a sliding menu opens up, allowing the user to choose the desired primitives, represented as game objects in Unity. These game objects can only interact if the user selects only two, both in 2D or 3D.

Once all the previous requirements, the Closest Distance game view displays the two game objects joined by a yellow line (Figure 6) that represents Minimal Distance from one to another. Additionally, an orange text that updates with every object's movement displays the Minimum Distance at the top of the screen.

For the Proximity Query game view, a similar displacement as the Closest Distance view is shown (Figure 7). However, since only the computation of the Proximity Query is necessary, the text at the top of the screen changes if the two objects in interaction are or not colliding. An additional element to the scene is that the objects change their color when colliding, allowing the user to identify a collision.

3.3.2 Special Features

Although these interactions between objects happen due to pure mathematical formulas adjusted to Unity specificities, some features were added to the game, allowing the user to interact with the chosen primitives. These features,

Table 4. Proximity Query Function Library.

Proximity Queries	
Contact Pair	**Inputs**
2D	
AABB-AABB	First AABB origin, Second AABB origin, First AABB length, Second AABB length, First AABB height, Second AABB height
OBB-OBB	First OBB normal vectors, Second OBB normal vectors, First OBB corners, Second OBB corners
3D	
Point-Sphere	Point position, Sphere origin, Sphere radius
Point-AABB	Point position, AABB origin, AABB length, AABB height, AABB width
Sphere-AABB	AABB origin, AABB length, AABB height, AABB width, Sphere origin, Sphere radius
Sphere-Sphere	First sphere origin, Second sphere origin, First sphere radius, Second Sphere radius
AABB-AABB	First AABB origin, Second AABB origin, First AABB length, Second AABB length, First AABB width, Second AABB width, First AABB height, Second AABB height
OBB-OBB	OBBs normal vectors, First OBB corners, Second OBB corners

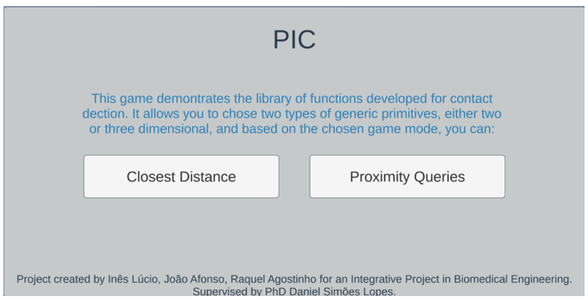

Figure 5. Initial menu of the developed game.

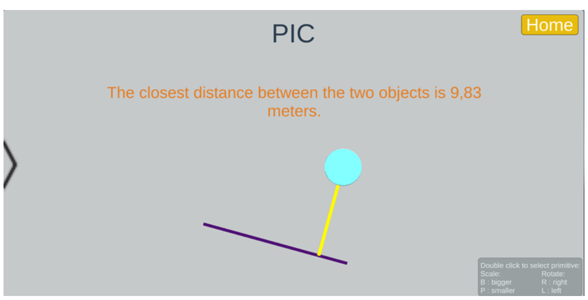

Figure 6. Closest distance calculations between a circle and a line in two dimensions.

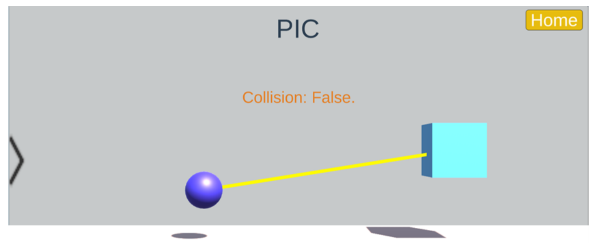

Figure 7. Collision detection between a sphere and an AABB in three dimensions.

represented in Figure 8, include scaling, rotation in specific directions, commanded by the keys "L"(left) and "R"(right), and movement on all axes.

3.4 Biomechanical Simulation

In addition to the game, we developed a simple biomechanical simulation of a robotic foot making contact with the ground using the provided libraries. The ground plane was represented by an AABB, and the foot was built using basic primitives such as spheres and AABBs, as shown in Figure 9. A sphere symbolizes the heel and toes of the foot, while an AABB represents the sole. The model aimed to simulate the phases of the gait cycle and to compute the Minimum Distances and collisions between the heel, sole, and toes with the ground.

Conclusion

The main objectives initially presented were achieved. We conducted an extensive literature review. Then, we choose a few contact pairs to implement in the function libraries. Given the extent of the objectives of this work, only simpler geometric primitives were implemented, and the computational cost of the computations used was not as high a priority as desired. Also, to demonstrate these libraries' features, we created a small game and a biomechanical simulation using Unity.

With the work developed, it is now possible for any individual to use the libraries developed for calculations of their interest. Besides the simulation developed, it would also be feasible to perform other biomechanical evaluations, such as hand-objects contact for rehabilitation applications or seat design assessments.

Hereafter, more complex primitives and functions can be added to the libraries. Furthermore, the libraries could be translated to other programming languages in order to extend their use. By continuing this work, when the libraries reach a considerable size and complexity, the final goal would be to publish them and share them with the community.

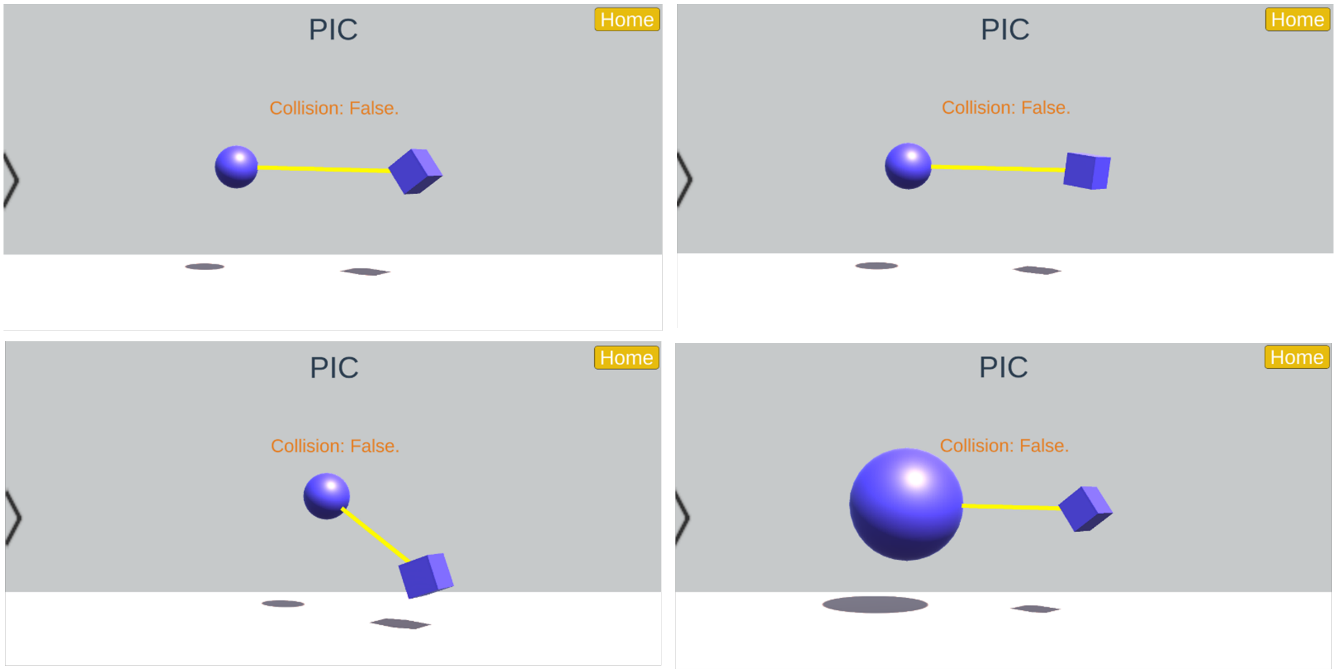

Figure 8. Collision detection between a sphere and an OBB (top left). The figure on the top right illustrates the rotation capacity of the objects. The figure on the bottom left illustrates the movement capacity of the objects. The figure on the bottom right illustrates the scaling capacity of the objects.

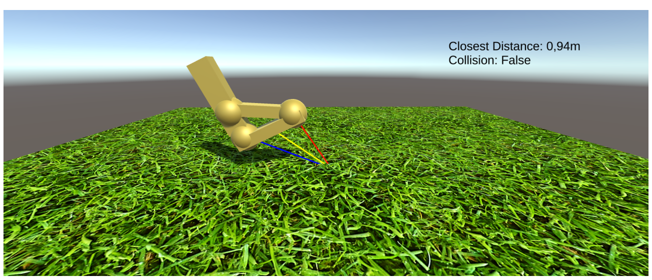

Figure 9. Biomechanical simulation of a robotic foot in contact with the ground.

Acknowledgments

A special thanks to Prof. Daniel Simões Lopes for having guided us throughout this work.

References

[1] Ming Lin, Dinesh Manocha, and Young Kim. *39 COLLISION AND PROXIMITY QUERIES.* 2017.

[2] Sheldon Andrews and Kenny Erleben. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21, New York, NY, USA, 2021. Association for Computing Machinery.

[3] D. S. Lopes, R. R. Neptune, J. A. Ambrósio, and M. T. Silva. A superellipsoid-plane model for simulating foot-ground contact during human gait. *Computer Methods in Biomechanics and Biomedical Engineering*, 19(9):954–963, 2016. PMID: 26325481.

[4] F. Wang, T. Poston, C.L. Teo, K.M. Lim, and E. Burdet. Multisensory learning cues using analytical collision detection between a needle and a tube. In *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings.*, pages 339–346, 2004.

[5] Various authors. Real time rendering, 2021.

[6] erincatto. Github - erincatto/box2d: Box2d is a 2d physics engine for games, Dec 2021.

[7] jslee02. Github - jslee02/awesome-collision-detection: A curated list of awesome collision detection libraries and resources, May 2022.

[8] Philip J Schneider and David H Eberly. *Geometric tools for computer graphics.* Morgan Kaufmann, 2003.

[9] Christer Ericson. *Real-time collision detection.* Elsevier [20]10, 2005.

[10] Euclidean distance, Jun 2022.

[11] Thomas Schwarzl. *2D game Collision Detection: An introduction to clashing geometry in games.* CreateSpace, 2012.

[12] Many MDM Contributors. 3d collision detection - game development: Mdn, Feb 2022.

Appendix I

Table 5. 2D Closest Distance.

Contact Pairs	References	Used References	Specific Location or Equation
2D			
Point-Point	Wikipedia Contributors, "Euclidean distance," Wikipedia, Jun. 19, 2022. https://en.wikipedia.org/wiki/Euclidean distance (accessed Jul. 12, 2022).		Section "Distance Formulas"
Point-Line	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Linear Component", "Point to Line". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 190-191.		
Point-Ray	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Linear Component", "Point to Ray". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 191-192.		
Point-Segment	1. Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Linear Component", "Point to Segment". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 192-193. 2. Ericson C., "Basic Primitive Tests", "Closest-point Computations", "Closest Point on Line Segment to Point ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 127-129		
Point-Polyline	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polyline". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 194-196.		
Point-Triangle	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polygon", "Point to Triangle". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 196-211.		
Point-Rectangle	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polygon", "Point to Rectangle". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 211-213.		
Point-Orthogonal-Frustum	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polygon", "Point to Orthogonal Frustum". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 213-216.		
Point-Convex-Polygon	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polygon", "Point to Convex Polygon". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 216-217.		
Point-Quadratic-Curve	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Quadratic Curve". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 217-219.		
Point-Polynomial-Curve	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Point to Polynomial Curve". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 219-221.		
Line-Line	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Linear Components", "Line to Line". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 221-222.		
Line-Ray	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Linear Components", "Line to Ray". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. P 222-223.		
Line-Line-Segment	Scheneider, J.P., Eberly, D. H., "Distance in 2D", "Linear Components", "Line to Segment". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 223-224.		

Table 5. 2D Closest Distance.

Contact Pairs	References	Used References	Specific Location or Equation
Ray-Ray	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Components”, “Ray to Ray”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 224-226.		
Ray-Line-Segment	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Components”, “Ray to Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 226-228.		
Line-Segment-Line-Segment	1. Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Components”, “Segment to Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 228-229. 2. Ericson, C., ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point of two line segments ”, “2D Segment Intersection”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 151-153		
Line-Segment-Triangle	Ericson, C., ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point of two line segments ”, “2D Segment Intersection”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 151-153		
Linear-Component-Polyline	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Component to Polyline or Polygon”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 229-230.		
Linear-Component-Polygon	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Component to Polyline or Polygon”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 229-230.		
Linear-Component-Quadratic-Curve	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Component to Quadratic Curve”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 231-233.		
Linear-Component-Polynomial-Curve	Scheneider, J.P., Eberly, D. H., “Distance in 2D”, “Linear Component to Polynomial Curve”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 233.		
AABB-Point	Ericson, C., ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point on AABB to Point ”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 130-132		
OBB-Point	Ericson, C., ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point on OBB to Point”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 132-134		
Triangle-Point	Ericson, C., ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point on Triangle to Point”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 136-142		

Table 6. 3D Closest Distance.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
3D			
Point-Point	Wikipedia Contributors, "Euclidean distance," Wikipedia, Jun. 19, 2022. https://en.wikipedia.org/wiki/Euclidean distance (accessed Jul. 12, 2022).		Section "Distance Formulas"
Point-Linear Component	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Linear Component". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 365-367		
Point-Line-Segment	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Linear Component", "Point to Ray or Line Segment". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 367-369.		
Point-Ray	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Linear Component", "Point to Ray or Line Segment". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 367-369.		
Point-Polyline	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Linear Component", "Point to Polyline". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 369-374.		
Point-Plane	1. Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Plane". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 374-376. 2. Christer Ericson, "Basic Primitive Tests", "Closest-point Computations", "Closest Point on Plane to Point ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 126-127		
Point-Triangle	1. Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Triangle". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 376-382. 2. sheldona, "contactFrictionSim/point_triangle_distance.cpp at main · sheldona/contactFrictionSim," GitHub, Jul. 29, 2021. https://github.com/sheldona/contactFrictionSim/blob/main/3rdParty/Discregrid/src/geometry/pointtriangledistance.cpp (Accessed May 23, 2022).		
Point-Rectangle	1. Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Rectangle". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 382-385. 2. Ericson, C, "Basic Primitive Tests", "Closest-point Computations", "Closest Point on 3D Rectangle to Point". Real-Time Collision Detection. Elsevier Inc., 2005. p. 135-136		
Point-Polygon	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Polygon". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 385-388.		
Point-Circle	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Circle or Disk". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 388-391.		
Point-Disk	Scheneider, J.P., Eberly, D. H., "Distance in 3D", "Point to Planar Component", "Point to Circle or Disk". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 388-391.		

Table 6. 3D Closest Distance.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Point-Tetrahedron	1. Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polyedron”, “General Problem”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 391-393. 2. Ericson, C, ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point Tetrahedron to Point”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 142-145		
Point-Convex Polyhedron	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polyedron”, “General Problem”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 393.		
Point-OBB	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polyedron”, “Point to Oriented Bounding Box”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 394-397.		
Point-Orthogonal Frustum	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polyedron”, “Point to Orthogonal Frustum”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 397-401.		
Point-Quadric Surface	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Quadric Surface”, “Point to General Quadric Surface”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 401-403.		
Point-Polynomial Curve	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polynomial Curve”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 405-407.		
Point-Polynomial Surface	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Point to Polynomial Surface”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 407-409.		
Line-Line	1. Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Lines and Lines”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 409-412. 2. Ericson, C, ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point of two lines ”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 146-147		
Line-Line-Segment	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Segment/Segment, Line/Ray, Line/Segment, Ray/Ray, Ray/Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 420-422.		
Line-Ray	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Segment/Segment, Line/Ray, Line/Segment, Ray/Ray, Ray/Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 418-420.		
Line-Segment-Line-Segment	1. Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Segment/Segment, Line/Ray, Line/Segment, Ray/Ray, Ray/Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 415-418. 2. Ericson, C, ”Basic Primitive Tests”, ”Closest-point Computations”, ”Closest Point of two line segments ”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 148-151		
Ray-Line-Segment	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Segment/Segment, Line/Ray, Line/Segment, Ray/Ray, Ray/Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 424-426.		

Table 6. 3D Closest Distance.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Ray-Ray	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Components”, “Segment/Segment, Line/Ray, Line/Segment, Ray/Ray, Ray/Segment”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 422-424.		
Linear Component-Triangle	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Component to Triangle, Rectangle, Tetrahedron, Oriented Box”, “Linear Component to Triangle”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 433-441.		
Linear Component-Rectangle	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Component to Triangle, Rectangle, Tetrahedron, Oriented Box”, “Linear Component to Rectangle”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 441-447.		
Linear Component-Tetrahedron	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Component to Triangle, Rectangle, Tetrahedron, Oriented Box”, “Linear Component to Tetrahedron”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 447-450.		
Linear Component-OBB	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Linear Component to Triangle, Rectangle, Tetraedron, Oriented Box”, “Linear Component to Oriented Bounding Box”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 450-465.		
Line-Quadric Surface	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Line to Quadric Surface”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 465-467.		
Line-Polynomial Surface	Scheneider, J.P., Eberly, D. H., “Distance in 3D”, “Line to Polynomial Surface”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 467-468.		

Table 7. 2D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
2D			
Point-Point	1. Schwarzl, T. "Collision Detection: Point-Point Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 29 2. jeffThompson, "CollisionDetection/CodeExamples/PointPoint at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Point-Line	1. Schwarzl, T. "Collision Detection: Point-Line Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 50 2. jeffThompson, "CollisionDetection/CodeExamples/LinePoint at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Point-Line-Segment	Schwarzl, T. "Collision Detection: Point-Line-Segment Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 51		
Point-Rectangle	1. Schwarzl, T. "Collision Detection: Rectangle-Point Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 43 2. jeffThompson, "CollisionDetection/CodeExamples/PointRect at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Point-Oriented-Rectangle	Schwarzl, T. "Collision Detection: Point-Oriented-Rectangle Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 52		
Point-Circle	1. Schwarzl, T. "Collision Detection: Circle-Point Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 38 2. jeffThompson, "CollisionDetection/CodeExamples/PointCircle at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Point-Polygon	jeffThompson, "CollisionDetection/CodeExamples/PolyPoint at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Point-Triangle	jeffThompson, "CollisionDetection/CodeExamples/TriPoint at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Line-Line	1. Schwarzl, T. "Collision Detection: Line-Line Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 30-31 2. jeffThompson, "CollisionDetection/CodeExamples/LineLine at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018 3. "Maths - Intersection of shapes - Martin Baker," Euclideanspace.com, 2022. 4. Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243, p 243-244.		
Line-Ray	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243.		
Line-Line-Segment	1. Schwarzl, T. "Collision Detection: Line-Line-Segment Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 53 2. Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243.		

Table 7. 2D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
Line-Rectangle	1. Schwarzl, T. "Collision Detection: Rectangle-Line Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 44 2. jeffThompson, "CollisionDetection/CodeExamples/LineRect at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018		
Line-Oriented-Rectangle	Schwarzl, T. "Collision Detection: Line-Oriented-Rectangle Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 54		
Line-Circle	1. Schwarzl, T. "Collision Detection: Circle-Line Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 39 2. jeffThompson, "CollisionDetection/CodeExamples/LineCircle at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018 3. Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and Circular Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.		
Line-Arc	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and Circular Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.		
Line-Quadratic Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and General Quadratic Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247.		
Line-Polynomial Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polynomial Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 248-255.		
Line-Polyline	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polylines". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.		
Line-Polygon	1. jeffThompson, "CollisionDetection/CodeExamples/PolyLine at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018 2. Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polylines". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.		
Line-Triangle	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Line Against Triangle ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 184-188		
Line-Quadrilateral	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Line Against Quadrilateral ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 188-190		
Ray-Ray	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243, p 243-244.		
Ray-Line-Segment	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243.		

Table 7. 2D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Ray-Polyline	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polylines". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.		
Ray-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polylines". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.		
Ray-Circle	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and Circular Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.		
Ray-Arc	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and Circular Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.		
Ray-Quadratic Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Quadratic Curves", "Linear Components and General Quadratic Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247.		
Ray-Polynomial Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components and Polynomial Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 248-255.		
Ray-Triangle	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Triangle ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 190-194		
Ray-Convex Polyhedron	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Convex Polyhedron ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 198-201		
Line-Segment-Line-Segment	1. Schwarzl, T. "Collision Detection: Line-Segment-Line-Segment Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 32-34 2. Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Linear Components". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243, 244-245.		
Line-Segment-Rectangle	Schwarzl, T. "Collision Detection: Rectangle-Line-Segment Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 45		
Line-Segment-Oriented-Rectangle	Schwarzl, T. "Collision Detection: Line-Segment-Oriented-Rectangle Collision". 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 55		

Table 7. 2D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Line-Segment-Circle	1. Schwarzl, T. “Collision Detection: Circle-Line-Segment Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 40 2. erincatto, “box2d/b2_collide_edge.cpp at main · erincatto/box2d,” GitHub, Jul. 11, 2020. https://github.com/erincatto/box2d/blob/main/src/collision/b2collideedge.cpp (Accessed May 24, 2022). 3. Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Quadratic Curves”, “Linear Components and Circular Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.	2.	Function b2CollideEdge AndCircle.
Line-Segment-Polyline	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Polylines”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.		
Line-Segment-Polygon	1. erincatto, “box2d/b2_collide_edge.cpp at main · erincatto/box2d,” GitHub, Jul. 11, 2020. https://github.com/erincatto/box2d/blob/main/src/collision/b2collideedge.cpp (accessed May 24, 2022). 2. Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Polylines”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 246.	1.	Function b2CollideEdge AndPolygon.
Line-Segment-Arc	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Quadratic Curves”, “Linear Components and Circular Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247-248.		
Line-Segment-Quadratic Curve	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Quadratic Curves”, “Linear Components and General Quadratic Curves”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 247.		
Line-Segment-Polynomial Curve	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components and Polynomial Curves”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 248-255.		
Line-Segment-Convex Polyhedron	Ericson, C, ”Basic Primitive Tests”, ”Intersecting Lines, Rays, and (Directed) Segments”, ”Intersecting Ray or Segment Against Convex Polyhedron ”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 198-201		
Interval-Interval	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Linear Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 241-243, 245.		
Rectangle-Rectangle	1. Schwarzl, T. “Collision Detection: Rectangle-Rectangle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 27 2. jeffThompson, “CollisionDetection/CodeExamples/RectRect at master · jeffThompson/CollisionDetection,” GitHub, Dec. 12, 2018 3. “Maths - Intersection of shapes - Martin Baker,” Euclideanspace.com, 2022. 4. Parberry, I. “The Collision Module”, “AABBs”, Introduction to Game Physics with Box2D. London: Crc Press. 2016. p. 215-216.		

Table 7. 2D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Rectangle-Oriented-Rectangle	Schwarzl, T. “Collision Detection: Rectangle-Oriented-Rectangle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 46		
Rectangle-Circle	1. Schwarzl, T. “Collision Detection: Circle-Rectangle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 41 2. jeffThompson, “CollisionDetection/CodeExamples/CircleRect at master · jeffThompson/CollisionDetection,” GitHub, Dec. 12, 2018		
Rectangle-Polygon	jeffThompson, “CollisionDetection/CodeExamples/PolyRect at master · jeffThompson/CollisionDetection,” GitHub, Dec. 12, 2018		
Oriented-Rectangle-Oriented-Rectangle	1. Schwarzl, T. “Collision Detection: Oriented-Rectangle-Oriented-Rectangle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 35-37 2. “Maths - Intersection of shapes - Martin Baker,” Euclideanspace.com, 2022.		
Oriented-Rectangle-Circle	Schwarzl, T. “Collision Detection: Circle-Oriented-Rectangle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 42		
Circle-Circle	1. Schwarzl, T. “Collision Detection: Circle-Circle Collision”. 2D Game Collision Detection: An introduction to clashing geometry in games. CreateSpace Independent Publishing Platform, 2012. p. 28 2. jeffThompson, “CollisionDetection/CodeExamples/CircleCircle at master · jeffThompson/CollisionDetection,” GitHub, Dec. 12, 2018 3. “Maths - Intersection of shapes - Martin Baker,” Euclideanspace.com, 2022. 4. Parberry, I. “Mathematics For Game Physics”, “Reflections on Reflection”, “Bouncing Balls”. Introduction to Game Physics with Box2D. London: Crc Press. 2016. pg. 32-36. 5. erincatto, “box2d/b2_collide_circle.cpp at main · erincatto/box2d,” GitHub, Dec. 27, 2019. https://github.com/erincatto/box2d/blob/main/src/collision/b2collide circle.cpp (accessed May 24, 2022). 6. Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Quadratic Curves”, “Circular Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 257-258.		Function b2CollideCircles.
Circle-Arc	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Quadratic Curves”, “Circular Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 257-258.		
Circle-Polygon	1. jeffThompson, “CollisionDetection/CodeExamples/PolyCircle at master · jeffThompson/CollisionDetection,” GitHub, Dec. 12, 2018 2. erincatto, “box2d/b2_collide_circle.cpp at main · erincatto/box2d,” GitHub, Dec. 27, 2019. https://github.com/erincatto/box2d/blob/main/src/collision/b2collide circle.cpp (Accessed May 24, 2022).		Function b2CollidePolygon AndCircle.
Arc–Arc	Scheneider, J.P., Eberly, D. H., “Intersection in 2D”, “Quadratic Curves”, “Circular Components”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 257-258.		

Table 7. 2D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
Polygon-Polygon	1. jeffThompson, "CollisionDetection/CodeExamples/PolyPoly at master · jeffThompson/CollisionDetection," GitHub, Dec. 12, 2018 2. erincatto, "box2d/b2_collide_polygon.cpp at main · erincatto/box2d," GitHub, Dec. 27, 2019. https://github.com/erincatto/box2d/blob/main/src/collision/b2collidepolygon.cpp (accessed May 24, 2022).		Function b2CollidePolygons.
Triangle-Triangle	1. "Maths - Intersection of shapes - Martin Baker," Euclideanspace.com, 2022. 2. Ling-yu, W. "A faster triangle-to-triangle intersection test algorithm," Computer Animation and Virtual Worlds, vol. 25, no. 5–6, pp. 553–559, Oct. 2013, doi: 10.1002/cav.1558.		
Ray or Line-Segment-Triangle	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Triangle ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 190-194		
Ray or Line-Segment-Convex Polyhedron	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Convex Polyhedron ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 198-201		
Quadratic Curve-Quadratic Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Quadratic Curves", "General Quadratic Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 255-257.		
Polynomial Curve-Polynomial Curve	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "Polynomial Curves". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 262-265.		
Convex Polygon-Convex Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 2D", "The Method of the Separating Axes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 265-273.		Pseudocode on page(s): 269, 270-271, 273.

Table 8. 3D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
3D			
Point-Sphere	"3D collision detection - Game development — MDN," Mozilla.org, Jul. 04, 2022. https://developer.mozilla.org/en-US/docs/Games/Techniques/3Dcollisiondetection (accessed Jul. 12, 2022).		Section "Point vs. sphere"
Point-AABB	"3D collision detection - Game development — MDN," Mozilla.org, Jul. 04, 2022. https://developer.mozilla.org/en-US/docs/Games/Techniques/3Dcollisiondetection (accessed Jul. 12, 2022).		Section "Point vs. AABB"
Line-Triangle	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Triangles". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 485-488.		Pseudocode on page(s) 488.
Line-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Polygons". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 488-491.		Pseudocode on page(s) 490-491.
Line-Disk	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Component and Disk". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 491-493.		Pseudocode on page(s) 492-493.
Line-Polyhedra	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 493-498.		Pseudocode on page(s) 496-497.
Line-Quadric Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "General Quadric Surfaces". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 499-501.		Pseudocode on page(s) 500-501.
Line-Sphere	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Sphere". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 501-503.		Pseudocode on page(s) 502-503.
Line-Ellipsoid	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and an Ellipsoid". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 505-506, 507.		Pseudocode on page(s) 502-503.
Line-Cylinder	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Cylinders". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 507-512.		Pseudocode on page(s) 510-512.
Line-Cone	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 512-519.		Pseudocode on page(s) 516-519.
Line-Polynomial Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polynomial Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 519-520.		

Table 8. 3D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
Line-AABB	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Axis-Aligned Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 626-630.		Pseudocode on page(s) 628-630.
Line-OBB	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Oriented Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 630-634.		Pseudocode on page(s) 632-634.
Ray-Triangle	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Triangles". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 485-488.		Pseudocode on page(s) 488.
Ray-Polygon	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Polygons". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 488-491. 2. Pereira, J. M., "Interseções", "Interseções de uma semirreta com um plano". Introdução à Computação Gráfica. FCA, 2018. p. 203-204	1.	Pseudocode on page(s) 490-491.
Ray-Disk	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Component and Disk". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 491-493.		Pseudocode on page(s) 492-493.
Ray-Polyhedra	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 493-498.		Pseudocode on page(s) 496-497.
Ray-Quadric Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "General Quadric Surfaces". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 499-501.		Pseudocode on page(s) 500-501.
Ray-Sphere	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Sphere". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 501-504. 2. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Sphere ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 177-179	1.	Pseudocode on page(s) 502-503.
Ray-Ellipsoid	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and an Ellipsoid". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 505-506, 507.		Pseudocode on page(s) 502-503.
Ray-Cylinder	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Cylinders". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 507-512. 2. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Cylinder ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 194-198	1.	Pseudocode on page(s) 510-512.
Ray-Cone	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 512-519.		Pseudocode on page(s) 516-519.

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Ray-Polynomial Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polynomial Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 519-520.		
Ray-AABB	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Axis-Aligned Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 626-630.		Pseudocode on page(s) 628-630.
Ray-OBB	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Oriented Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 630-634. 2. Pereira, J. M., "Interseções", "Interseções de uma semirreta com um plano". Introdução à Computação Gráfica. FCA, 2018. p. 204-205 3. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Box ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 179-184	1.	Pseudocode on page(s) 632-634.
Line-Segment-Plane	Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Segment Against Plane ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 175-177		
Line Segment-Triangle	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Triangles". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 485-488.		Pseudocode on page(s) 488.
Line Segment-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Polygons". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 488-491.		Pseudocode on page(s) 490-491.
Line Segment-Disk	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Component and Disk". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 491-493.		Pseudocode on page(s) 492-493.
Line Segment-Polyhedra	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 493-498.		Pseudocode on page(s) 496-497.
Line Segment-Quadric Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "General Quadric Surfaces". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 499-501.		Pseudocode on page(s) 500-501.
Line Segment-Sphere	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Sphere". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 501-504. 2. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Sphere ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 177-179	1.	Pseudocode on page(s) 502-503.
Line Segment-Ellipsoid	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and an Ellipsoid". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 505-506, 507.		Pseudocode on page(s) 502-503.

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Line Segment-Cylinder	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and Cylinders". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 507-512. 2. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Cylinder ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 194-198	1.	Pseudocode on page(s) 510-512.
Line Segment-Cone	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Quadric Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 512-519.		Pseudocode on page(s) 516-519.
Line Segment-Polynomial Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Polynomial Surfaces", "Linear Components and a Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 519-520.		
Line Segment-AABB	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Axis-Aligned Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 626-630.		Pseudocode on page(s) 628-630.
Line Segment-OBB	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Linear Component and Oriented Bounding Box". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 630-634. 2. Ericson, C, "Basic Primitive Tests", "Intersecting Lines, Rays, and (Directed) Segments", "Intersecting Ray or Segment Against Box ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 179-184	1.	Pseudocode on page(s) 632-634.
Plane-Point	sheldona, "contactFrictionSim/CollisionDetect.cpp at main · sheldona/contactFrictionSim," GitHub, 2022. https://github.com/sheldona/contactFrictionSim/blob/main/src/collision/CollisionDetect.cpp (Accessed May 23, 2022).		Function collisionDetectPoint-Plane.
Plane-Line	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Planes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 482-484.		Pseudocode on page(s) 484.
Plane-Ray	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Planes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 482-484. 2. Pereira, J. M., "Interseções", "Interseções de uma semirreta com um plano". Introdução à Computação Gráfica. FCA, 2018. p. 202-203	1.	Pseudocode on page(s) 484.
Plane-Line Segment	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Linear Components and Planar Components", "Linear Components and Planes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 482-484.		Pseudocode on page(s) 484.
Plane-Plane	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components", "Two Planes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 529-531.		Pseudocode on page(s) 531.
Plane-Triangle	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components", "Triangle and Plane". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 534-539.		Pseudocode on page(s) 535-539.

Table 8. 3D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
Plane-Polyhedron with triangular faces	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Polyhedra”, “Trimeshes”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 543-544.		
Plane-Polygon	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Polyhedra”, “General Polyhedra”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 545.		
Plane-Polyhedron	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Polyhedra”, “General Polyhedra”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 546.		
Plane-General Quadric Surface	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Quadric Surfaces”, “Plane and General Quadric Surface”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 547-548.		
Plane-Cone	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Quadric Surfaces”, “Plane and Cone”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 563-582.		Pseudocode on page(s) 575, 576, 580-581, 582.
Plane-Polynomial Surface	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Polynomial Surfaces”, “The Algorithm”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 592-595.		Pseudocode on page(s) 592-593, 594.
Plane-AABB	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Plane and Axis-Aligned Bounding Box”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 634-635.		Pseudocode on page(s) 635.
Plane-OBB	1. Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Plane and Oriented Bounding Box”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 635-637. 2. Ericson, C. “Bounding Volumes”, “Basic Primitive Tests”, “Testing Primitives”,”Testing Box Against Plane”. Real-Time1 Collision Detection. Elsevier Inc., 2005. p. 161-164		
AABB-AABB	1. Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Axis-Aligned Bounding Boxes”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 637-639. 2. Ericson, C., “Bounding Volumes”, “Axis-aligned Bounding Boxes (AABBs)”, “AABB-AABB Intersection”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 79-80	1.	Pseudocode on page(s) 638.
AABB-Triangle	Ericson, C. “Bounding Volumes”, “Basic Primitive Tests”, “Testing Primitives”,”Testing AABB Against Triangle ”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 169-172		
AABB-Plane	sheldona, “contactFrictionSim/CollisionDetect.cpp at main · sheldona/contactFrictionSim,” GitHub, 2022. https://github.com/sheldona/contactFrictionSim/blob/main/src/collision/CollisionDetect.cpp (Accessed May 23, 2022).		Function collisionDetectBox-Plane.

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
OBB-OBB	1. Ericson, C., "Bounding Volumes", "Oriented Bounding Boxes (OBBs)", "OBB-OBB Intersection". Real-Time Collision Detection. Elsevier Inc., 2005. p. 101-106 2. Andrews, S., Erleben, K.. 2021. "Contact Generation", "Analytical Shapes", "OBB-OBB Intersection". Contact and friction simulation for computer graphics, p. 44-47. In ACM SIGGRAPH 2021 Courses (SIGGRAPH '21). Association for Computing Machinery, New York, NY, USA, Article 2, 1–124. https://doi.org/10.1145/3450508.3464571 3. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Miscellaneous", "Oriented Bounding Boxes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 639-644. 4. bepu, "bepuphysics1/BoxBoxCollider.cs at master · bepu/bepuphysics1," GitHub, Feb. 12, 2021. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/BoxBoxCollider.cs (Accessed May 26, 2022). 5. RandyGaul, "qu3e/q3Collide.cpp at master · RandyGaul/qu3e," GitHub, Feb. 14, 2020. https://github.com/RandyGaul/qu3e/blob/master/src/collision/q3Collide.cpp (accessed May 26, 2022).		
Sphere-swept Volume	Ericson, C., "Bounding Volumes", "Sphere-swept Volumes", "Sphere-swept Volume Intersection". Real-Time Collision Detection. Elsevier Inc., 2005. p. 114-115		
Triangle-Triangle	1. Ericson, C. "Bounding Volumes", "Basic Primitive Tests", "Testing Primitives","Testing Triangle Against Triangle". Real-Time Collision Detection. Elsevier Inc., 2005. p. 172-175. 2. bepu, "bepuphysics1/TriangleTrianglePairTester.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/TriangleTrianglePairTester.cs (Accessed May 26, 2022). 3. Devillers, O. and Guigue, P., "Faster Triangle-Triangle Intersection Tests. RR-4488, INRIA," 2002. [Online]. Available: https://hal.inria.fr/inria-00072100/document 4. Möller, T. "A Fast Triangle-Triangle Intersection Test," Journal of Graphics Tools, vol. 2, no. 2, pp. 25–30, Jan. 1997, doi: 10.1080/10867651.1997.10487472. 5. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components", "Triangle and Triangle". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 539-542.		
Triangle-Polyhedron with triangular faces	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polyhedra", "Trimeshes". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 543-544.		
Triangle-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polyhedra", "General Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 545-546.		

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Triangle-Cone	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Quadric Surfaces", "Triangle and Cone". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 583-587.		Pseudocode on page(s) 584.
Triangle-Polynomial Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polynomial Surfaces", "The Algorithm". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 592-595.		Pseudocode on page(s) 592-593, 594.
Triangle-Convex Polyhedron	bepu, "bepuphysics1/TriangleConvexPairTester.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/TriangleConvexPairTester.cs (Accessed May 26, 2022).		
Polyhedron-Triangle	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polyhedra", "General Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 546.		
Polyhedron-Disk	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polyhedra", "General Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 546.		
Polyhedron-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polyhedra", "General Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 546.		
Polynomial Surface-Disk	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polynomial Surfaces", "The Algorithm". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 592-595.		Pseudocode on page(s) 592-593, 594.
Polynomial Surface-Polygon	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Planar Components and Polynomial Surfaces", "The Algorithm". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 592-595.		Pseudocode on page(s) 592-593, 594.
Quadric Surface-Quadric Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Quadric Surfaces", "General Intersection". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 596-604.		
Polynomial Surface-Polynomial Surface	Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "Polynomial Surfaces", "Ellipsoids". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 608-611.		

Table 8. 3D Proximity Query.

Contact Pairs	**References**	**Used Refer-ence(s)**	**Specific Location or Equation**
Convex Polyhedra-Convex Polyhedra	1. Scheneider, J.P., Eberly, D. H., "Intersection in 3D", "The Method of Separating Axes", "Separation of Stationary Convex Polyhedra". Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 611-. 2. bepu, "bepuphysics1/GeneralConvexPairTester.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/GeneralConvexPairTester.cs (Accessed May 26, 2022). 3. bepu, "bepuphysics1/MPRToolbox.cs at master · bepu/bepuphysics1," GitHub, Feb. 19, 2015. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/MPRToolbox.cs (Accessed May 26, 2022). 4. bepu, "bepuphysics1/MinkowskiToolbox.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/MinkowskiToolbox.cs (Accessed May 26, 2022). 5. DanielChappuis, "reactphysics3d/ConvexPolyhedronVsConvexPolyhedronAlgorithm.cpp at master · DanielChappuis/reactphysics3d," GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/ConvexPolyhedronVsConvexPolyhedronAlgorithm.cpp (Accessed May 26, 2022).	1.	Pseudocode on page(s) 612, 613-614, 614-615.

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Sphere-Sphere	1. “3D Theory - Collision Detect - Martin Baker,” Euclideanspace.com, 2022. 2. Pereira, J.M. “Interseções: Interseções de duas esferas”. Introdução à Computação Gráfica. FCA, 2018. p. 200-202 3. Ericson, C. “Bounding Volumes”, “Spheres”, “Sphere-Sphere intersection”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 88-89 4. Andrews, S., Erleben, K.. 2021. “Contact Generation”, “Analytical Shapes”, “Sphere-Sphere Intersection”. Contact and friction simulation for computer graphics, p. 42. In ACM SIGGRAPH 2021 Courses (SIGGRAPH ’21). Association for Computing Machinery, New York, NY, USA, Article 2, 1–124. https://doi.org/10.1145/3450508.3464571 5. sheldona, “contactFrictionSim/CollisionDetect.cpp at main · sheldona/contactFrictionSim,” GitHub, 2022. https://github.com/sheldona/contactFrictionSim/blob/main/src/collision/CollisionDetect.cpp (Accessed May 23, 2022). 6. bepu, “bepuphysics1/SphereTester.cs at master · bepu/bepuphysics1,” GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/SphereTester.cs (Accessed May 26, 2022). 7. DanielChappuis, “reactphysics3d/SphereVsSphereAlgorithm.cpp at master · DanielChappuis/reactphysics3d,” GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/SphereVsSphereAlgorithm.cpp (Accessed May 26, 2022).	5.	Function collisionDetectSphereSphere.
Sphere-Plane	1. Ericson, C. “Bounding Volumes”, “Basic Primitive Tests”, “Testing Primitives”,”Testing Sphere Against Plane”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 160-161 2. Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Quadric Surfaces”, “Plane and Sphere”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 548-551.	2.	Pseudocode on page(s) 550-551.
Sphere-AABB	1. Ericson, C. “Bounding Volumes”, “Basic Primitive Tests”, “Testing Primitives”,”Testing Sphere Against AABB”. Real-Time Collision Detection. Elsevier Inc., 2005. p. 165-166 2. sheldona, “contactFrictionSim/CollisionDetect.cpp at main · sheldona/contactFrictionSim,” GitHub, 2022. https://github.com/sheldona/contactFrictionSim/blob/main/src/collision/CollisionDetect.cpp (Accessed May 23, 2022). 3. Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Sphere and Axis-Aligned Bounding Box”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 644-646.	2. and 3.	2. Function collisionDetectSphereBox. 3. Pseudocode on page(s) 645.

Table 8. 3D Proximity Query.

Contact Pairs	**References**	**Used Reference(s)**	**Specific Location or Equation**
Sphere-OBB	1. Ericson, C. "Bounding Volumes", "Basic Primitive Tests", "Testing Primitives","Testing Sphere Against OBB". Real-Time Collision Detection. Elsevier Inc., 2005. p. 166-167 2. Andrews, S., Erleben, K.. 2021. "Contact Generation", "Analytical Shapes", "Sphere-OBB Intersection". Contact and friction simulation for computer graphics, p. 42-44. In ACM SIGGRAPH 2021 Courses (SIGGRAPH '21). Association for Computing Machinery, New York, NY, USA, Article 2, 1–124. https://doi.org/10.1145/3450508.3464571 3. bepu, "bepuphysics1/BoxSphereTester.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/BoxSphereTester.cs (Accessed May 26, 2022).		
Sphere-Triangle	1. Ericson, C. "Bounding Volumes", "Basic Primitive Tests", "Testing Primitives","Testing Sphere against Triangle ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 167-168 2. bepu, "bepuphysics1/TriangleSpherePairTester.cs at master · bepu/bepuphysics1," GitHub, 2022. https://github.com/bepu/bepuphysics1/blob/master/BEPUphysics/CollisionTests/CollisionAlgorithms/TriangleSpherePairTester.cs (Accessed May 26, 2022).		
Sphere-Capsule	DanielChappuis, "reactphysics3d/SphereVsCapsuleAlgorithm.cpp at master · DanielChappuis/reactphysics3d," GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/SphereVsCapsuleAlgorithm.cpp (Accessed May 26, 2022).		
Sphere-Polygon	Ericson, C. "Bounding Volumes", "Basic Primitive Tests", "Testing Primitives","Testing Sphere Against Polygon ". Real-Time Collision Detection. Elsevier Inc., 2005. p. 168-169		
Sphere-Convex Polyhedron	DanielChappuis, "reactphysics3d/SphereVsConvexPolyhedron Algorithm.cpp at master · DanielChappuis/reactphysics3d," GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/SphereVsConvexPolyhedronAlgorithm.cpp (Accessed May 26, 2022).		
Capsule-Capsule	DanielChappuis, "reactphysics3d/CapsuleVsCapsuleAlgorithm.cpp at master · DanielChappuis/reactphysics3d," GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/CapsuleVsCapsuleAlgorithm.cpp (Accessed May 26, 2022).		
Capsule-Convex Polyhedron	DanielChappuis, "reactphysics3d/CapsuleVsConvexPolyhedron Algorithm.cpp at master · DanielChappuis/reactphysics3d," GitHub, 2022. https://github.com/DanielChappuis/reactphysics3d/blob/master/src/collision/narrowphase/CapsuleVsConvexPolyhedronAlgorithm.cpp (Accessed May 26, 2022).		

Table 8. 3D Proximity Query.

Contact Pairs	References	Used Reference(s)	Specific Location or Equation
Cylinder-Plane	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Planar Components and Quadric Surfaces”, “Plane and Cylinder”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 551-563.		Pseudocode on page(s) 562-563.
Torus-Line	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Linear Component and Torus”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 659-662.		
Torus-Ray	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Linear Component and Torus”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 659-662.		
Torus-Line-Segment	Scheneider, J.P., Eberly, D. H., “Intersection in 3D”, “Miscellaneous”, “Linear Component and Torus”. Geometric Tools for Computer Graphics. Morgan Kaufamann Publishers. 2003. p 659-662.		